Progress on General Capacity Using Network Coding

Muriel Médard
Joint with Jay Kumar Sudararajan, MinJi Kim, Devavrat Shah, Ralf Koetter
Progress on General Capacity Using Network Coding

**Achievement:** for networks of depth one (like switch)

Provide a systematic way of characterizing achievable region and benefit of coding

**How it works:**

- An edge between two vertices if the two configuration cause conflict (i.e., they cannot be served simultaneously)

**Assumptions and limitations:**

- Approach is general, but speedup characterized for depth one
- Works in separable settings
- Algorithms are centralized

**Status quo**

- When separation holds, what is the benefit of having network coding?
- Major difficulty I: in non-multicast settings, codes are an open problem
- Major difficulty II: time-varying nature of traffic and of network operation, e.g., changing codes
- Major difficulty III: even without coding, performance is ill understood
- State of the art I: pick a system (say COPE) and run experimental trials to demonstrate improvement
- State of the art II: pick a multicast example and work it out by hand

- Fixing family of possible codes, give systematic representation of achievable region using conflict graph representation
- Obviates the need for finding clever schedules by hand
- Difficulty of problem now becomes one of characteristics of conflict graph (for instance, perfection)—it is a combinatorial, graph-theoretic question
- Finding schedules now comes from conflict graph

**End-of-phase goal**

To obtain general results for networks with multiple layers

To incorporate MAC constraints into the conflict graphs, allowing mixture of MAC and scheduling

To provide a set of systematic approaches to determine schedules

Create online schemes, in the flavor of i-slip, to trade-off complexity and effectiveness of schedules, with possible decentralization

**New insight/intellectual tool**

- Can bound speed-up using imperfection ratio of conflict graph

**Community challenge**

- How can we approximate difficult capacity region problems?
- How can we create schedules from such approximations?
- What is the loss that comes from a distributed scheduling?

**Ramification:** bring problem to its combinatorial essence, which determines difficulty
Status Quo

- When separation holds, what is the benefit of having network coding?
- In non-multicast settings, codes are an open problem
- Time-varying nature of traffic and of network operation, e.g. changing codes
- Even without coding, performance is ill understood
- State of the art:
  - Pick a system (say COPE) and run experimental trials to demonstrate improvement
  - Pick a multicast example and work it out by hand
• Transform scheduling problem into a combinatorial graph-theoretic question
• Fix family of possible codes; give systematic representation of achievable region using conflict graphs
• Obviates the need for finding clever schedules by hand
• Difficulty of problem now depends on the characteristics of conflict graph (for instance, perfection)
• Finding schedules now comes from conflict graph
Achievement

• Focus on networks of depth one (e.g. multicast switch)

• Provide a systematic way of characterizing the achievable rate region and the benefit of coding – using a simple and intuitive graph theoretic formulation

• Scheduling algorithms using rate decomposition approach
• 4 x 3 switch
• Traffic: mix of unicast and special pattern
• Used randomized variant of max weight scheduling
How it works

- Conflict graph:
  - A vertex for every network state *(i.e. switch configuration)*
  - An edge between two vertices if the two states cause conflict *(i.e. two flows that cannot be served simultaneously)*

- Map valid service configurations of the system of queues to stable sets of conflict graph
How it works

- Achievable rate region is the stable set polytope of the conflict graph
  - Closed form characterization known for several, but not all, classes of graphs
- Computing schedules based on rate decomposition – maps to weighted coloring of the graph
  - Polynomial time algorithm if the graph is perfect
How it works

Speedup needed for 100% throughput

\[ = \min\{t \mid A \subseteq t\cdot\text{NC}\} \]

• Admissible region:
  – Set of rates such that no input or output node is overloaded
  – Corresponds to the clique inequality polytope of conflict graph

• Can use switch speedup to “expand” the rate region

• Speedup needed for 100% throughput = Factor of expansion for network coding region to cover admissible region

• Result: Imperfection ratio of conflict graph is an upper bound on the speedup needed for 100% throughput
Assumptions and limitations

• Approach is general, but the speedup results are only for a network of depth one

• Works in settings where separation holds

• The algorithms are centralized. This could be a limitation some scenarios
End of phase goals

• To obtain general results for networks with multiple layers
• To incorporate MAC constraints into the conflict graphs, allowing mixture of MAC and scheduling
• To provide a set of systematic approaches to determine schedules
• Create online schemes, in the flavor of i-slip, to trade-off complexity and effectiveness of schedules, with possible decentralization
Community Challenge

• How can we approximate difficult capacity region problems?
• How can we create schedules from such approximations?
• What is the loss that comes from a distributed scheduling?
Conclusions and Discussion

• New systematic approach to scheduling and rate region questions in network coding

• Transforms problem into a combinatorial graph theoretic formulation

• Incorporates time sharing among different connection states and codes – scheduling using graph coloring
Summary: Progress on General Capacity Using Network Coding

**Achievement:** for networks of depth one (like switch)

Provide a systematic way of characterizing achievable region and benefit of coding

**How it works:**

An edge between two vertices if the two configuration cause conflict

(i.e. They cannot be served simultaneously)

\[
\min \left\{ t : A \subseteq t \text{ NC} \right\} = \frac{2}{3}
\]

Speedup needed for 100% throughput

Can bound speed-up using imperfection ratio of conflict graph

**Assumptions and limitations:**

- Approach is general, but speedup characterized for depth one
- Works in separable settings
- Algorithms are centralized

**Status quo**

- When separation holds, what is the benefit of having network coding?
- Major difficulty I: in non-multicast settings, codes are an open problem
- Major difficulty II: time-varying nature of traffic and of network operation, e.g. changing codes
- Major difficulty III: even without coding, performance is ill understood
- State of the art I: pick a system (say COPE) and run experimental trials to demonstrate improvement
- State of the art II: pick a multicast example and work it out by hand

**New insight/intellectual tool**

Fixing family of possible codes, give systematic representation of achievable region using conflict graph representation

- Obviates the need for finding clever schedules by hand
- Difficulty of problem now becomes one of characteristics of conflict graph (for instance, perfection)—it is a combinatorial, graph-theoretic question
- Finding schedules now comes from conflict graph

**End-of-phase goal**

To obtain general results for networks with multiple layers

To incorporate MAC constraints into the conflict graphs, allowing mixture of MAC and scheduling

To provide a set of systematic approaches to determine schedules

Create online schemes, in the flavor of i-slip, to trade-off complexity and effectiveness of schedules, with possible decentralization

**Community challenge**

How can we approximate difficult capacity region problems?

How can we create schedules from such approximations?

What is the loss that comes from a distributed scheduling?

**Ramification:** bring problem to its combinatorial essence, which determines difficulty